GemStone/S 64 Bit™
Release Notes

Version 3.6.4

June 2022

INTELLECTUAL PROPERTY OWNERSHIP

This documentation is furnished for informational use only and is subject to change without notice. GemTalk Systems LLC assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation.

Warning: This computer program and its documentation are protected by copyright law and international treaties. Any unauthorized copying or distribution of this program, its documentation, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted under the maximum extent possible under the law.

The software installed in accordance with this documentation is copyrighted and licensed by GemTalk Systems under separate license agreement. This software may only be used pursuant to the terms and conditions of such license agreement. Any other use may be a violation of law.

Use, duplication, or disclosure by the Government is subject to restrictions set forth in the Commercial Software - Restricted Rights clause at 52.227-19 of the Federal Acquisitions Regulations (48 CFR 52.227-19) except that the government agency shall not have the right to disclose this software to support service contractors or their subcontractors without the prior written consent of GemTalk Systems.

This software is provided by GemTalk Systems LLC and contributors "as is" and any expressed or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall GemTalk Systems LLC or any contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

COPYRIGHTS

PATENTS

GemStone software is or has been covered by U.S. Patent Number 6,256,637 "Transactional virtual machine architecture" (1998-2018), Patent Number 6,360,219 "Object queues with concurrent updating" (1998-2018), Patent Number 6,567,905 "Generational garbage collector with persistent object cache" (2001-2021), and Patent Number 6,681,226 "Selective pessimistic locking for a concurrently updateable database" (2001-2021).

TRADEMARKS

GemTalk, **GemStone**, **GemBuilder**, **GemConnect**, and the GemTalk logo are trademarks of GemTalk Systems LLC, or of VMware, Inc., previously of GemStone Systems, Inc., in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Solaris, **Java**, and **Oracle** are trademarks or registered trademarks of Oracle and/or its affiliates. **SPARC** is a registered trademark of SPARC International, Inc.

Intel and **Pentium** are registered trademarks of Intel Corporation in the United States and other countries.

Microsoft, **Windows**, and **Windows Server** are registered trademarks of Microsoft Corporation in the United States and other countries.

Linux is a registered trademark of Linus Torvalds and others.

Red Hat and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

Ubuntu is a registered trademark of Canonical Ltd., Inc., in the U.S. and other countries.

SUSE is a registered trademark of Novell, Inc. in the United States and other countries.

AIX, **POWER7**, **POWER8**, **POWER9** and **VisualAge** are trademarks or registered trademarks of International Business Machines Corporation.

Apple, **Mac**, **MacOS**, and **Macintosh** are trademarks of Apple Inc., in the United States and other countries.

CINCOM, **Cincom Smalltalk**, and **VisualWorks** are trademarks or registered trademarks of Cincom Systems, Inc.

Raspberry Pi is a trademark of the Raspberry Pi Foundation.

RabbitMQ is a trademark of VMware, Inc. in the U.S. and other countries.

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective owners. Trademark specifications are subject to change without notice. GemTalk Systems cannot attest to the accuracy of all trademark information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

GemTalk Systems LLC
15220 NW Greenbrier Parkway
Suite 240
Beaverton, OR 97006

Preface

About This Documentation

These release notes describe changes in the GemStone/S 64 Bit™ version 3.6.4 release. Read these release notes carefully before you begin installation, upgrade, or development with this release.

No separate Installation Guide is provided with this release. For instructions on installing GemStone/S 64 Bit version 3.6.4, or upgrading or converting from previous products or versions, see the Installation Guide for version 3.6.2.

Terminology Conventions

The term "GemStone" is used to refer to the server products GemStone/S 64 Bit and GemStone/S, and the GemStone family of products; the GemStone Smalltalk programming language; and may also be used to refer to the company, now GemTalk Systems LLC, previously GemStone Systems, Inc. and a division of VMware, Inc.

Technical Support

Support Website

gemtalksystems.com

GemTalk's website provides a variety of resources to help you use GemTalk products:

- **Documentation** for the current and for previous released versions of all GemTalk products, in PDF form.
- **Product download** for the current and selected recent versions of GemTalk software.

- **Bugnotes**, identifying performance issues or error conditions that you may encounter when using a GemTalk product.
- **Supplemental Documentation** and **TechTips**, providing information and instructions that are not in the regular documentation.
- **Compatibility matrices**, listing supported platforms for GemTalk product versions.

We recommend checking this site on a regular basis for the latest updates.

Help Requests

GemTalk Technical Support is limited to customers with current support contracts. Requests for technical assistance may be submitted online (including by email), or by telephone. We recommend you use telephone contact only for urgent requests that require immediate evaluation, such as a production system down. The support website is the preferred way to contact Technical Support.

Website: techsupport.gemtalksystems.com

Email: techsupport@gemtalksystems.com

Telephone: (800) 243-4772 or (503) 766-4702

Please include the following, in addition to a description of the issue:

- The versions of GemStone/S 64 Bit and of all related GemTalk products, and of any other related products, such as client Smalltalk products, and the operating system and version you are using.
- Exact error message received, if any, including log files and statmonitor data if appropriate.

Technical Support is available from 8am to 5pm Pacific Time, Monday through Friday, excluding GemTalk holidays.

24x7 Emergency Technical Support

GemTalk offers, at an additional charge, 24x7 emergency technical support. This support entitles customers to contact us 24 hours a day, 7 days a week, 365 days a year, for issues impacting a production system. For more details, contact GemTalk Support Renewals.

Training and Consulting

GemTalk Professional Services provide consulting to help you succeed with GemStone products. Training for GemStone/S is available at your location, and training courses are offered periodically at our offices in Beaverton, Oregon. Contact GemTalk Professional Services for more details or to obtain consulting services.

Table of Contents

Chapter 1. Release Notes for 3.6.4

Chapter

1 Release Notes for 3.6.4

Overview

GemStone/S 64 Bit™ 3.6.4 is a new version of the GemStone/S 64 Bit object server. Version 3.6.4 adds SuperDoit scripting, FFI support for structs by value, and other feature enhancements and bug fixes. We recommend everyone using or planning to use GemStone/S 64 Bit upgrade to this new version.

These Release Notes include changes between the previous version of GemStone/S 64 Bit, v3.6.3, and v3.6.4. If you are upgrading from a version prior to 3.6.3, review the release notes for each intermediate release to see the full set of changes.

The Installation Guide has not been updated for this release. For installation, upgrade and conversion instructions, use the Installation Guide for version 3.6.2.

Supported Platforms

Platforms for Version 3.6.4

GemStone/S 64 Bit version 3.6.4 is supported on the following platforms:

- Red Hat Enterprise Linux Server, CentOS Linux, and Rocky Linux 7.9 and 8.5; and Ubuntu 18.04 and 20.04
  GemStone performs testing on a mixture of Red Hat, CentOS, and Rocky servers; these are all fully certified platforms. Any reference to Red Hat applies to all these distributions.
- Solaris 10 on x86
- AIX 7.1 and 7.2
- OSX 11.1 (Big Sur) with Darwin 20.2.0 kernel on x86, and OSX 10.15.6 (Catalina) with Darwin 19.6.0 kernel; and OSX 11.6 (Big Sur) with Darwin 20.6.0 kernel on Apple M1. v3.6.4 is known to work on MacOS Monterey.
  (Mac is supported for development only)

For more information and detailed requirements for each supported platforms, please refer to the *GemStone/S 64 Bit Installation Guide* for that platform.

GemBuilder for Smalltalk (GBS) Versions

The following versions of GBS can be used with GemStone/S 64 Bit version 3.6.4:

GBS/VW version 8.6

VisualWorks 9.1.1 32-bit and 64-bit
▸ Windows 10 ▸ RedHat ES 7.9 and 8.5; Ubuntu 18.04 and 20.04

GBS/VA version 5.4.6

VAST Platform 11.0.0	VAST Platform 10.0.2	VA Smalltalk 9.2.2	VA Smalltalk 8.6.3
▸ Windows Server 2016 and Windows 10	▸ Windows Server 2016 and Windows 10	▸ Windows Server 2016 and Windows 10	▸ Windows Server 2016 and Windows 10

For more details on GBS and client Smalltalk platforms and requirements, see the *GemBuilder for Smalltalk Installation Guide* for that version of GBS.

VSD Version

The GemStone/S 64 Bit v3.6.4 distribution includes VSD version 5.5.3. The previous version of GemStone/S 64 Bit, v3.6.3, included VSD v5.5.2. VSD 5.5.3 includes improved searching, bug fixes, and updates to topaz online help.

Note that in GemStone/S 64 Bit v3.6 and later, **statmonitor** writes additional information to the statmonitor file. As a result, statmonitor files from v3.6 and later cannot be read by versions of VSD earlier than v5.5. VSD 5.5.3 can read statmonitor files generated in older versions of GemStone/S 64, 32-bit GemStone, and GBS, as well as those generated by GemStone/S 64 Bit v3.6 and later.

VSD v5.5.3 is included with the GemStone distribution, and can also be downloaded as a separate product. For details or to download, go to https://gemtalksystems.com/vsd/.

Open Source Library Versions

The version of OpenSSL has been updated to 1.1.1n.

1. Changes in this version

Distribution Changes

Rowan

There is a new top level directory, `$GEMSTONE/rowan`, containing scripts and upgrade material for users of Rowan code management. Rowan is an open-source code management project that is in a limited preview status, and undergoing active development.

The `$GEMSTONE/bin` directory now includes `extent0.rowan.dbf`.

SuperDoit

As described under "SuperDoit scripting feature", starting on page 17, the SuperDoIt feature includes added scripts in $GEMSTONE/bin, and a directory containing examples, `$GEMSTONE/examples/superDoit`.

SuperDoit is implemented using Rowan; however, SuperDoit scripts are fully supported in base GemStone, and creating and executing SuperDoit scripts requires no Rowan knowledge.

The SuperDoit API includes support for Rowan-specific functions. These are not documented in these Release Notes; see `$GEMSTONE/examples/GsCommand` for examples.

$GEMSTONE/upgrade

Starting with v3.6, Rowan code has been included in the distribution, under $GEMSTONE/upgrade/projects. This directory is required to be present for some functions, including superDoit solo scripts.

The location of the projects directory within the distribution may change in a future release.

$GEMSTONE/seaside directory no longer owner-write

The directory $GEMSTONE/seaside had installation permissions that incorrectly allowed owner write; this permission has been removed.

Support for Rowan upgrades

This release includes upgrade support for applications running certain earlier versions of GemStone 3.6.x with Rowan versions 1.2.x and 2.2.x, that are upgrading to v3.6.4.

Banner notice when running with SELinux

SELinux is a RedHat mode that applies additional security restrictions. When running in this mode, the process log banners now will include a notice; for example,

```
| MEMORY: 64187 MB , getauxval(AT_SECURE) = 2 |
```

Additional information in stone log messages

Stone log messages now includes additional information, such as the IP addresses of Gem sessions, to help with diagnosing problems.

Print GCI trace records on protocol error

To aid in analysis of GCI protocol errors, recent GCI call history is now automatically printed to the gem log when a GCI protocol error occurs. GCI call trace records are maintained in a wraparound buffer, and the most recent 100 records will be printed to the log.

Rolling over Gem log files

For long-lived Gems, it can be helpful to roll over the log file periodically. This can now be done using the following new method.

`System class >> startNewGemLog:` *aFileName*
Close the existing log file for this Gem and start a new log with the name *aFileName*. *aFileName* must be a file name, not a path ('/' is not allowed); this file name is appended to the result of `System gemLogPath` to produce the new log file. If a file named *aFileName* already exists in this location, it will be opened for append. If the file cannot be created or written to, this method returns an error and the existing log file will continue to be used. Has no effect and signals a Warning in a topaz -l or other linked GCI application.

GsHostProcess changes

Using files for stdin, stdout, and stderr

GsHostProcess has been substantially modified to allow stdin, stdout, and stderr to be specified as files. The following methods have been added:

`GsHostProcess >> stderrPath`
`GsHostProcess >> stderrPath:` *aStringOrUtf8*
`GsHostProcess >> stdinPath`
`GsHostProcess >> stdinPath:` *aStringOrUtf8*
`GsHostProcess >> stdoutPath`
`GsHostProcess >> stdoutPath:` *aStringOrUtf8*

`GsHostProcess >> appendToFiles:` *aBoolean*
If *aBoolean* == true and out or err are paths for files to be opened, causes already existing files to be opened for append.

Added public instance API

The following instance methods have been added, similar to the equivalent class methods. These allow modification of the GsHostProcess instance before execute or fork.

`GsHostProcess >> commandLine:` *aString*
Set the command line that will be executed

`GsHostProcess >> execute`
Execute the code in the instance variable commandLine.

`GsHostProcess >> executeWithInput:` *stdinString*
Execute the code in the instance variable commandLine, writing *stdinString* to the stdin of the child.

`GsHostProcess >> fork`
Forks the child process specified by the instVars.

Removed private method _fork:

The method `GsHostProcess >> _fork:` has been removed; it is replaced by `GsHostProcess >> fork:`.

fork and fork: requirements for standard files

When using `GsHostProcess >> fork` or `GsHostProcess >> fork:` directly, you should specify one or more of `GsHostProcess>>stderrPath:`, `GsHostProcess>>stdinPath:`, and `GsHostProcess>>stdoutPath:`, to specify where the child should go for its standard files. Otherwise the child may hang when trying to read or write to a pipe connected to the parent process, if the parent is not executing the data read and writes implemented in `GsHostProcess>>_executeWithInput:`.

Upgrade impact of these changes

GsHostProcess definition had added two new instance variables. This can create issues with customer subclasses or GsHostProcess and persistent instances of GsHostProcess.

During upgrade, the existing GsHostProcess class definition will be renamed ObsoleteGsHostProcess, and moved to ObsoleteClasses. GsHostProcess has a new reserved OOP in v3.6.4. Any subclasses of or persistent instances of GsHostProcess in your 3.6.3 or earlier application will be subclasses of or instances of ObsoleteGsHostProcess after upgrade to v3.6.4.

Subclasses of GsHostProcess should be recompiled to be subclasses of the new GsHostProcess class (Globals at: #GsHostProcess), and methods modified as necessary and recompiled.

GsSingleRefPathFinder disallowed in remote sessions

It is now disallowed to find reference paths using GsSingleRefPathFinder in sessions that are remote from the Stone (on a remote cache).

GsObjectInventory profiling methods now wait up to 60 seconds for the GcLock

GsObjectInventory scans previously only waited for 3 seconds for another session using the GcLock to finish and release the lock. Now, these will wait for 60 seconds before timing out.

Private methods handling days from 1970 removed

The following private methods were removed

```
Date >> _asDaysFrom1970
Date class >> _newFromDays1970:
```

GsFileIn now ignores login, logout

GsFileIn allows programmatic filein of topaz-formatted files. GsFileIn now ignores **login** and **logout** commands that are encountered in the input file.

Error changes

The Smalltalk symbol #errIllegalRamSelf has been added for the error ERR_illegalRamSelf #2156, with the error message 'Self is not a ram oop, method needs recompile.'

ProfMonitor object creation tree report format

This report was unreadable with deep trees. The indents have been reduced, and indents more than 25 deep are all printed with the same indent level.

External Session Changes

GsExternalSession and GciLegacyExteranlSession previously hardcoded errors to GciError or GciLegacyError; now, methods invoke a new method `GsExternalSession >> gciErrorClass`, to return the class of Error.

The following method was added:

```
GsTsExternalSession >> isReadReady
```
Return true if the session is ready to read a result or error.

OtherPassword split with ReadOtherUserProfile

Finding out any information about another user's UserProfile, such as the lastLoginTime or authenticationScheme, previously required OtherPassword privilege. This privilege also allowed modifying another users's UserProfile.

To allow reading another user's UserProfile, but not allow modification, a new privilege, ReadOtherUserProfile, has been added. A UserProfile with this privilege can read specific information for other UserProfiles, but cannot make changes.

A UserProfile with OtherPassword has implicitly also ReadOtherUserProfile, so there is no need to grant this additional privilege to users with OtherPassword.

The following methods can be executed by a UserProfile with OtherPassword, ReadOtherUserProfile, or both privileges:

```
UserProfile >> activeUserIdLimit
UserProfile >> authenticationScheme
UserProfile >> hasLoginLogging
UserProfile >> isReadOnly
UserProfile >> lastLoginTime
UserProfile >> lastPasswordChange
UserProfile >> ldapBaseDn
UserProfile >> ldapSearchFilterDn
UserProfile >> loginsAllowedBeforeExpiration
UserProfile >> passwordAgeLimit
UserProfile >> passwordAgeWarning
UserProfile >> staleAccountAgeLimit
UserProfile >> userIdAlias
UserProfile >> passwordNeverExpires
```

Private removed method

The method `UserProfile >> _validateUserProfileAccess` has been removed.

Other Added Methods

`Array >> copyNotNilFrom:` *startIndex* `to:` *stopIndex*
Return an Array containing the non-nil elements that are within the specified offsets of the receiver.

`Array >> indexOfNotNil:` *startOffset* `to:` *endOffset*
If *startOffset* <= *endOffset*, returns the first offset within in the specified range of a non-nil element of the receiver. If *startOffset* > *endOffset*, returns the last offset within the specified range of a non-nil element of the receiver. Returns zero if all are nil. Intended for use only on Arrays of size <= 2000; performance on larger arrays will be slow.

`GsFile class >> stderrServer`
Returns an instance of the receiver which can be used to write to the standard error of the current Gem process, or nil if an error occurs.

`GsTestCase >> assert:` *objA* `includes:` *objB*

`GsTestCase >> assert:` *objA* `includesIdentical:` *objB*

`GsTestCase >> assert:` *objA* `isKindOfClass:` *objB*

`GsTestCase >> deny:` *objA* `includes:` *objB*

`GsTestCase >> deny:` *objA* `includesIdentical:` *objB*

`GsTestCase >> isArm64`

Topaz changes

Multiple -C arguments accepted

Previously, if separate -C arguments were passed to topaz, only the final one was applied; earlier -C arguments were ignored. To pass in multiple configuration parameters, they needed to be appended within a single -C argument string.

Now, you may include the -C argument multiple times, and each -C argument will be applied in the order given on the command line.

DISPLAY OOPS is now on by default

Previously, **display oops** was **off** by default. The oops of objects are often key information for resolving issues or bugs; to avoid cases in which this information was missing in customer logs, the default is now **on**. For example, for a command such as:

```
topaz 1> exec 'abc' %
```

in previous versions, this printed:

```
abc
```

in v3.6.4:

```
[39747073 size:3  String] abc
```

This can be overridden to omit the extra information by entering **omit oops** at the topaz command line, or in .topazini. **display classoops** remains off by default.

Last argument of -L, -l, -r now applied

-L, -l and -r are mutually exclusive arguments that determine the library to load, and if the login will be linked or RPC. Previously, while you could include multiple of this set, if either -l or -L was used, the login would be linked. Now, the final appearance on the command line of any of this set takes precedence.

EDIT command additional options

The EDIT command now accepts method specifications directly, rather than requiring an initial SET CLASS. The following options are available:

EDIT *classname* >> *selectorSpec*
: edit the given instance method.

EDIT *classname* CLASS >> *selectorSpec*
: edit the given class method.

EDIT LAST
: edit the code entered by the last **run**, **printit**, **doit**, **method:** or **classmethod:** command.

Cache statistics changes

Added Statistics

The following cache statistic has been added, as part of the changes for bug #49771, described on page 29.

ExtentGrowFailedTotal (Stn)
: Number of times an attempt to grow an extent failed.

The following statistics has also been added:

AvailableMemoryKB (Linux_System)
: An estimate of how many kilobytes of memory is available for starting new processes without triggering system swapping.

Change in units for GcHighWaterPage

As of 3.6.4, the value reported for GcHighWaterPage for scanning repository is reported in units of 0.1%; a value of 1000 means no scan is in progress.

2. SuperDoit scripting feature

A new scripting environment has been added, to allow you to easily write command line scripts entirely in Smalltalk, without needing a shell script. SuperDoit scripting provides a framework for command line arguments and usage sections, as well as other features to support writing complex scripts.

SuperDoit requires access to specific files in the distribution. This includes code that is currently in the `$GEMSTONE/upgrade` directory. To execute SuperDoit scripts,

- $GEMSTONE must be defined
- `$GEMSTONE/bin` must be on your search path
- `$GEMSTONE/upgrade` directory must be present.
- For stone scripts, `$GEMSTONE/version.txt` must be present.

SuperDoits have not been tested with X509-secured GemStone.

Script Types

SuperDoit scripts come in two variants:

- **stone**
  executes the script as a normal login to a running Stone. Login details are provided in a `.topazini`, which may be provided on the command line using the **-I** argument, or in one of the standard `.topazini` locations: the current working directory, or the home directory of the user executing the script.

  All the details required for a login to a Stone in your specific environment must be available, in the `.topazini`, $GEMSTONE_NRS_ALL, and/or other environment variables.

  By convention, these scripts end in `.stone`.

- **solo**
  executes the script as a solo session. Commits, and operations that require a stone, are not allowed, and solo scripts cannot login RPC. Solo scripts by default use `$GEMSTONE/bin/extent0.rowan.dbf` for their solo extent.

  Solo scripts do not automatically look for a standard `.topazini` file. Solo scripts use a solo-specific topaz initialization file, located within the superDoit project, which logs in as SystemUser. You may use the **-I** argument to specify a different topaz initialization file.

  By convention, solo scripts end in `.solo`.

Script structure

A SuperDoit script consists of a number of sections, each one started by a keyword, and ending with %. All sections are optional.

The Smalltalk code that the script executed is in a section with the keyword **doit**. Note that while this looks similar to topaz syntax, it is a keyword defining the code execution section; topaz syntax is not understood in solo and stone scripts.

A minimal SuperDoit script might be, for example,

```
#!/usr/bin/env superdoit_solo
doit
   Time now asString
%
```

In addition to a doit section, there are number of sections that you may include depending on your script's requirements. Most commonly, you will want to include sections for script command line options, in the options section, and usage information in a usage section.

An example of a script with the options, usage, and doit sections:

```
#!/usr/bin/env superdoit_stone
# Print the contents of DbfHistory with a byte limit
options
   {SuperDoitOptionalOptionWithRequiredArg
     long: 'bytes' short: 'b'}
%
usage
   $basename [-h] [-D] [-b <number>] [-- [-I <topazini>]]
   Return DbfHistory for the stone specified by <topazini>, or
   if not specified, the default .topazini for the environment.
   The optional -b argument specifies to return only the first
   <number> bytes of the DbfHistory.
%
doit
   self bytes isNil
      ifTrue: [DbfHistory]
      ifFalse: [DbfHistory copyFrom: 1 to:
         (self bytes asInteger min: DbfHistory size)]
%
```

This script show how access to the argument is handled: `self bytes` returns the contents of the command line argument specified using `-b` or `--bytes`.

Also note that the usage here is simplified; the arguments `--help`, `--Debug`, and `--bytes` may also be used on the command line. See the section under "Arguments" on page 19.

Finally, the argument to topazini that is specifically mentioned in the usage is only one of a number of arguments to topaz that can be passed to the SuperDoit script and forwarded to topaz. These arguments, if used, are separated from the script arguments by --. See the section "Arguments to Topaz" on page 20.

Commonly Used Sections

doit
holds the body of the script

options
specifies command line arguments; see the possible specifications under "Arguments" on page 19.

usage
provide usage text. This can be in any format; all text between the usage and closing % is printed in response to the -h or --help argument to the script.

input
specifies code (topaz or gs) that is filed in prior to executing the doit.

customoptions
Allows you to override the default arguments for help and debug.

Complex Scripting

The following sections allow functionality to be broken down into supporting methods and associated state.

method
Define a method on the script itself

instvars
Define instance variables for the script, which can be accessed by the methods defined in method, method: and classmethod: sections.

method:
Define an instance method on an existing class.

classmethod:
Define a class method on an existing class.

Rowan Support

The following are advanced sections to support Rowan repositories:

projectshome
Declare the value of the ROWAN_PROJECTS_HOME environment variable.

specs
Specify an array of Rowan load specification STON objects used to load external projects into the image.

specurls
Specify a list of spec urls that reference the location of a Rowan load specification STON object.

Script Comments

You may add comments to a superDoit script by prefixing the line with #.

These comments can be put in anywhere within the script.

Arguments

Command line arguments may be defined using name-value pairs, names without values, or as positional arguments.

The options section defines the named and name-value command line arguments; positional arguments are accessed in the doit section, sending self positionalArgument to return the array of zero or more positional arguments that were included on the command line.

Named and name-value pairs can be specified either using getopts syntax *-Character value* (referred to as "short" in SuperDoit), or *--String=value* ("long").

The long version is always supported; you may also specify a short version when defining the script arguments.

Arguments with values may be required or optional, and optional arguments may have a default value specified. See the specifications on "Defining Script Arguments" on page 20.

Standard arguments

By default, all scripts have the help and debug options:

-h or --help
: Returns usage information specified in the usage section

-D or --Debug
: If an exception occurs, the script stops in topaz rather than exiting. This allows you to debug using topaz commands such as where.

To exclude help and/or debug options, you can define a section in your script for customoptions.

Note that since the help section is processed and returned in Smalltalk code, executing a script to return the help text requires a successful login to the Stone or solo extent.

Defining Script Arguments

The script options are specified as an array of objects (in GemStone Array Builder syntax, curly-brace delimited and period separated), specified using the following expressions.

```
SuperDoitOptionalOptionWithNoArg long:
SuperDoitOptionalOptionWithNoArg long:short:
SuperDoitOptionalOptionWithRequiredArg long:
SuperDoitOptionalOptionWithRequiredArg long:default:
SuperDoitOptionalOptionWithRequiredArg long:short:
SuperDoitOptionalOptionWithRequiredArg long:short:default:
SuperDoitRequiredOptionWithRequiredArg long:
SuperDoitRequiredOptionWithRequiredArg long:short:
```

For example, to specify an argument that may be supplied either with `-b` *value*, with `--bytes=`*value*, or omitted:

```
options
   {SuperDoitOptionalOptionWithRequiredArg
     long: 'bytes' short: 'b'}
%
```

Arguments to Topaz

In addition to your script-specific arguments, you may also include arguments that are directives to topaz. These are included following a -- on the command line.

For example, to pass in both a script argument and a topazini:

```
getDbfHistory.stone -b 100 -- -I .topazini
```

This can be used to pass in any other topaz argument, for example to override the default rowan extent to execute a superDoit script:

```
simple.solo -- -C GEM_SOLO_EXTENT=$GEMSTONE/bin/extent0.dbf
```

Most topaz directives can be used, and will override those specified by the SuperDoit infrastructure or the configured environment. Any use of the topaz -S option is ignored, and solo scripts may only log in linked.

Usage

The usage section includes a section of text that will be displayed when the script is invoked with **-h** or **--help**.

```
usage
   $basename [-h] [-D] [-b <number>] [-- [-I <topazini>]]
   Return DbfHistory for the stone specified by <topazini>, or
   if not specified, the default .topazini for the environment.
   The optional -b argument specifies to return only the first
   <number> bytes of the DbfHistory.
%
```

The usage section is optional. If it is not included in the script, when **-h** or **-help** is invoked, the default help with **-h/--help/-D/--debug** options is displayed.

Coding in the doit section

The doit section consists of GemStone Smalltalk code. Within the doit, self corresponds to the instance of SuperDoitExecution. This provides access to the script arguments and environment, and set of convenience helper methods.

Available methods include (but are not limited to):

`executionStoneName`
: Returns the name of the running stone; for solo, this will be gs64stone.

`globalNamed:` *globalName*
: Return a global variable with the given name.

`globalNamed:` *globalName* `ifAbsent:` *notfoundblock*
: Return a global variable with the given name, executing the *notfoundblock* if it does not exist.

`positionalArgs`
: Returns an array containing all positional arguments from the command line.

`stderr`
: The stderr that the script writes to.

`stdout`
: The stdout that the script writes to.

`noResult`
: At the end of the script, this allows you to return with no messages on the command line.

`exit:` *errmsg* `withStatus:` *anInteger*
: Exit the script, writing *errmsg* to stderr and with the OS process exit status set to *anInteger.*

`exitWithStatus:` *anInteger*
: Exits the script, with the OS process exit status set to *anInteger.*

`isSolo`
Return true if in a solo session.

`log:` anObject
Write the argument in STON format to stdout.

`logErrorMessage:` *aString*
Write the given string on stderr.

`logMessage:` *aString*
Write the given string on stdout.

`commandLine`
Returns the command line for the script.

Examples and Templates

$GEMSTONE/examples/superDoit/

This directory contains a number of example scripts, which can be executed as is, and script templates, illustrating the use of arguments and error handling.

Rowan-specific GsCommands examples

The scripts in $GEMSTONE/examples/GsCommands/ show the use of the rowan GsCommand package with superDoit scripts, illustrating the use of the Rowan-specific sections supported by superDoit scripts.

3. FFI Changes

The FFI has a number of additional features and bug fixes in this version.

CByteArray support for copy

Previously, #copy was disallowed for instances of kinds of CByteArray. This is now supported.

In addition to copy semantics, as implemented by `Object >> copy`, the following method has been added:

`CByteArray >> shallowCopy`
: Returns a copy of the receiver which shares the receiver's instance variables. If self autoRelease == true, the result's C memory will be a copy of the C memory of the receiver, with result autoRelease == true. Otherwise the result will contain a reference to the C memory of the receiver, with result autoRelease == false.

CHeader support for variable arguments

While the FFI supported C functions with variable arguments in previous releases, now the instance methods generated by CHeader include a separate keyword, `varArgs:`, allowing you to pass in variable arguments. The argument to `varArgs:` should be an Array in which there are two entries for each variable argument: the first is the type symbol, and the second is the value.

The earlier syntax is still valid; FFI wrapper classes generated in earlier versions of GemStone do not need to be regenerated after upgrade to v3.6.4, and message sends to the previously generated methods do not need to be updated.

Improved access to errno

Previously, to fetch error information, the method `CCallout class >> errno` and `errno:` accessed the errno information from the GsProcess. There was a risk that two interleaved FFI calls (within a single GsProcess) could access this, and set/retrieve incorrect values.

Added method callWith:errno:

The method `CCallout >> callWith:errno:`, a variant of `CCallout >> callWith:`, has been added, to avoid a risk of incorrect errno information. Calls to `CCallout class >> errno` and `CCallout class >> errno:` should be replaced by this call.

The `errno:` argument to `CCallout >> callWith:errno:` takes nil, or an array with at least one slot; the first element of this array may be nil or a SmallInteger. The array must be modifiable; that is, do not use a literal array.

- If the `errno:` argument is nil, it is not expected that errno information will be needed; however, the GsProcess's errno is still set, and you may still invoke #errno.
- If the `errno:` argument is an array and the first element is a SmallInteger (most commonly, zero), errno will be set to this value before the call is executed.
- If the `errno:` argument is an array, on return from the function, the first element of the array will be set to an SmallInteger errno that was set by the function execution.

Support for C functions with struct type arguments or results

The FFI now supports functions with C structs that are passed by value for arguments, and functions that return a C struct by value. Structs passed by address is not the same as a struct passed by value; these were previously supported by CPointers.

To support these calls, the class CCalloutStructs has been added, and is generated by CHeader wrapper methods when the source C header files includes struct arguments or return values.

A struct passed by value is represented as a kind of CByteArray, which must be of the correct size.

It is recommended to use `CHeader>>wrapper*` methods to generate the instances of CCalloutStructs and subclasses of CByteArray, to ensure that the CCalloutStructs and CByteArray are correctly constructed.

Limitations:

- Use of CCalloutStructs is supported only on Linux and MacOS on x86_64 and ARM/Apple M1 systems. Instances of CCalloutStructs generated by parsing the header files are platform-independent, but can only be used to make C calls on supported platforms.
- If the C function takes variable arguments, the variable argument values may not be structures passed by value.
- Structs by value are not supported as arguments nor as results, if any other argument to the C function is a C double or C float. However, structs that are passed as argument or results may themselves have fields of type double or float.

Added class CCalloutStructs

The class CCalloutStructs is a new subclass of CCallout. An instance of CCalloutStructs represents the information needed to call a C function using the FFI, where one or more of the result or arguments to the function are C structs (or C++ classes), passed by value.

The value of an argument or result that is a struct must be a kind of CByteArray with the correct size; it is strongly recommended to create a subclass of CByteArray using `CHeader>>wrapperForTypeNamed:`.

With functions that use structs, it is recommended to use CHeader to generate methods as well as data structure classes. Creating instance of CCalloutStructs requires using lower level protocol, including the struct sizes array.

Invoking these hand-crafted CCalloutStructs is done with the method `CCalloutStructs >> callWith:structResult:errno:`.

- If the function does not return a struct, but has struct arguments, the `structResult:` argument should be nil. You may use `callWith:` or `callWith:errno:`.
- If the function returns a struct, the `structResult:` argument must be an instance of a kind of CByteArray. The sender must allocate this CByteArray with the correct size before passing to the `structResult:` keyword.

Instances of CCalloutStructs have an instance variable structSizes, which is an array of the byte sizes of the return value, followed by the byte sizes of arguments. If the return value

is not a struct, the first element is nil; for any arguments that are not structs, the corresponding slot in the array will be nil.

CHeader generated wrapper classes

When generating a wrapper class using `CHeader >>wrapperNamed:*`, the resulting class will have instance methods that invoke either a CCallout or a CCalloutStructs instance, depending on the use of structs.

- If the function returns a struct, the instance method will invoke an instance of CCalloutStructs. The instance method will include an additional argument (in the first position).

  This first argument must be invoked with an allocated instance of a kind of CByteArray appropriate for the function's return struct type.
- If the function has a struct argument, but does not return a struct, the instance method will invoke an instance of CCalloutStructs. The instance method will only have arguments keywords for the actual arguments.
- If the function does not have struct arguments nor returns a struct, the instance method will invoke an instance of CCallout, as in previous releases.

Example of struct return and variable arguments

The following example shows both a hand-crafted and a CHeader wrapper construction for a call that returns a struct. This example is based on the open source RabbitMQ® messaging library. Note that for simplicity, the login example uses `(CPointer newNull)` for the argument that requires an established connection from an earlier call.

Example 1.1 Example with structs and varArgs using RabbitMQ

"create wrapper for the struct"

```
| wrapper |
wrapper := (CHeader path: '/usr/include/amqp.h')
      wrapperForTypeNamed: 'amqp_rpc_reply_t'.
UserGlobals at: wrapper name put: wrapper.
```

"call login function via hand-constructed CCalloutStructs. Note that the argument arrays for the `args:` keyword, `structSizes:`, and `callWith:` have different sizes, as they include or omit the result and the variable arguments)."

```
| theResult  myCO fArgs varArgs |
theResult := amqp_rpc_reply_t new.
myCO := CCalloutStructs
   library: (CLibrary named: '$GEMSTONE/testlib/testmq')
   name: 'amqp_login'
   result:  #struct
   args: #( #'ptr' #'const char*' #'int32' #'int32' #'int32'
      #'int32' )
   varArgsAfter: 6
   structSizes: { (theResult size) . nil . nil . nil . nil .
      nil . nil ).
```

```
fArgs := { (CPointer newNull) . '/'  . 0 . 131072 . 0 . 0  }.
varArgs := #( #'const char*' 'guest' #'const char*' 'guest' ).
myCO
   callWith: (fArgs, varArgs)
   structResult: theResult
   errno: nil.
theResult
```

"create wrapper for the library, with just the `amqp_login` function"

```
| class wrapper |
wrapper := (CHeader path: '/usr/include/amqp.h')
   wrapperNamed: 'MyAmq'
   forLibraryAt: '$GEMSTONE/testlib/testmq'
   select: [:each | each name = 'amqp_login'].
wrapper initializeFunctions.
UserGlobals at: wrapper name put: wrapper.
```

"call login function via instance method on wrapper. The instance of `amqp_rpc_reply_t` that is provided as an argument is the return value in `theResult`."

```
| theResult |
theResult := MyAmq new
   amqp_login_: (amqp_rpc_reply_t new)
   _: (CPointer newNull)_: '/'_: 0 _: 131072 _: 0 _: 0
   varArgs: #( #'const char*' 'guest' #'const char*' 'guest' ).
theResult
```

Other Image Changes related to FFI

Added Methods

The following methods have been added to support CCalloutStructs and other FFI changes:

`Boolean >> asBit`
Returns 1 if self==true, 0 otherwise.

`CPointer class >> forAddress:` *anInteger*
Create a new instance for the given address. *anInteger* must be representable as an unsigned 64bit integer.

`CByteArray >> byteArrayFrom:` *zeroBasedStart* `numBytes:` *anInteger*
Return a new ByteArray containing the specified bytes of the receiver.

`CByteArray >> byteArrayFromCharStarAt:` *zeroBasedOffset* `numBytes:` *anInteger*
Using the 8 bytes starting at *zeroBasedOffset* as a char * type, return a ByteArray with specified number of bytes.

`CByteArray >> pointerAt:` *zeroBasedOffset* `resultClass:` *aClass* `numBytes:` aSize
Returns an instance of *aClass* encapsulating the C pointer fetched from 8 bytes of the receiver starting at *zeroBasedOffset*. *aClass* must be CByteArray, CPointer, a subclass of CByteArray or a subclass of CPointer.

`CByteArray >> stringFrom:` *zeroBasedStart* `numBytes:` *anInteger*
Return a new String containing the specified bytes of the receiver.

`CByteArray >> stringFromCharStarAt:` *zeroBasedOffset* `numBytes:` *anInteger*
Using the 8 bytes starting at *zeroBasedOffset* as a char * type, return a String containing specified number bytes without regard to NUL bytes

`CByteArray >> utf16FromPointerAt:` *zeroBasedOffset* `numBytes:` *anInteger*
Using the 8 bytes starting at *zeroBasedOffset* as a ushort * type, return a Utf16 with specified number of bytes.

`CByteArray >> utf8FromCharStarAt:` *zeroBasedOffset* `numBytes:` *anInteger*
Using the 8 bytes starting at *zeroBasedOffset* as a char * type, return a Uf8 with specified number of bytes.

`CDeclaration >> byteSizeForMalloc`
Return a size rounded up to a multiple of 8 bytes.

`CDeclaration >> containsFloat`

`CDeclaration >> isStruct`

Removed methods

`CByteArray >> _signed:at:with:`

`CDeclaration >> baseAccessorForOffset:`

`CDeclaration >> baseStructByteSize`

`CDeclaration >> baseUnionByteSize`

`CDeclaration >> byteSizeRounded`

`CHeader >> initializeFunctionsCodeForFunctions:libraryPathExpressionString:`

CDeclaration refactored

Both the public and private API of CDeclaration has been extensively reworked.

Customers using this class directly may have to update their application. The CDeclaration instance variables were reordered; as a result, Repository upgrade creates a new version of this class.

4. Bugs Fixed

The following bugs in v3.6.3 are fixed in v3.6.4.

Risk of SPC corruption for allInstances and related repository scan operations

Repository scan operations that invoke primitive 1030 to return results as GsBitmap instances (`allInstances*`, `allObjects*`, `allReferences*`; see the bugnote for a complete list), has a timing bug in acquiring the GcLock. The Gem executing this primitive could get invalid page views, which may result in cache corruption. (#49945)

Reclaim issues

Crash for pageId 0 in pages needing reclaim or scavenge

If the ReclaimGem encounters pageId 0 on pages needing to be scavenged or on the list of reclaim page to reclaim, it crashed. This was unnecessary; this kind of unexpected state can be ignored, as is currently done for other types of unexpected state. (#49893, #49883)

Other potential unnecessary crashes

The code has been reviewed for other cases in which an internal state error produces a guarantee failure, causes the process to crash and dump core. These are now handled in a less aggressive way, allowing the system to continue operation. (#49896)

Reclaim problems under certain cases of heavy load

A number of improvements has been made in Reclaim to correctly handle some specific cases of heavy load, in which reclaim was not able to keep up.

Cases have been seen where reclaim could be paused due to "Cache starved"; reclaim to stop while there were still unreclaimed dead; and the ReclaimGem's transaction state could become incorrect such that it does not respond to sigAborts quickly enough, which caused spikes in commit record count. (#49964).

Symbol garbage collection may collect a referenced Symbol

A code path was found in which a referenced Symbol could be garbage collected. (#49886)

Gem crash recovery may cause Shrpcmon to SEGV

When a Gem crashes, the Shrpcmon performs recovery to clean up spin locks. It was possible for the address into the process table to be out of range, causing the Shrpcmon to crash. (#49988)

File descriptor leak in Gem

On login, a number of file descriptors are needed. One of these file descriptors was not released on logout. There was also a file descriptor that was opened that is not necessary. (#49983)

objectAudit issue in DataPageCheck not handled well

If objectAudit detects a problem with DataPageCheck, it some recent versions it may have produced large error sets, or caused the audit to crash. (#49830)

pageaudit may fail if Stone startup takes too long

The pageaudit utility starts a temporary Stone, and if the Stone startup process takes too long, the pageaudit may timeout and fail with an error. (#49875)

GsDevKit upgrade bugs fixed

There were bugs in the code to upgrade GsDevKit (issues 15, 16, 21 and 23 on the github project). These fixes are included in the distribution in $GEMSTONE/examples/seaside/bin/GsDevKit_upgrade.gs. (#49795, 48510)

Cache Statistics Issues

Extent grow cache statistics issues

The following stone cache statistics were not updated in some cases of grows initiated internally. The affected statistics are:

```
totalExtentGrows
timeInExtentGrows
avgTimePerExtentGrow
```

The statistic `ExtentGrowFailedTotal` has been added, to track extent grow failures; see ExtentGrowFailedTotal (Stn) Number of times an attempt to grow an extent failed. on page 16.

(#49771)

FreeOops not reliable

The Stone statistics **FreeOops** was sometimes set to an incorrect value. (#49986)

.topazini could silently clear -X arguments for X509 login

By default, topaz automatically reads a existing .topazini file on startup. If the .topazini sets any of the standard login parameters (gemstone, username, etc.), it clears any X509 login parameters, including those passed in via the topaz command line. This design is intended to avoid mixing sets of login parameters

Now, a warning is printed if command line -X arguments are cleared by a .topazini setting. (#49761)

Time comparison incorrect results for arguments in same millisecond range

The comparison methods > < = converted the operands to milliseconds, and this returned incorrect results in cases where the operands's milliseconds values matched; incorrectly returning true for = and incorrectly returning false for < or >. (#49798)

During tranlog restore, existing file without permissions not reported

If tranlog restore encountered a transaction log that it did not have permission to read, it stopped restore, as if restore was complete, without reporting a permission error. (#49837)

Login possible during restore

It was possible to login during a restore from backup, which caused the restore to fail. (#49958)

Commit issues during suspendCommitsForFailover

When commits are suspended for failover (System suspendCommitsForFailover), an attempt to commit will error. The commit attempt is not aborted, and an attempt to abort or resumeCommits causes the session to lose its connection to the stone. (#49943)

If the SymbolGem needs to commit, this will fail; the Stone will continue to attempt to restart the SymbolGem which will continue to fail to commit. (#49944).

SecurityErrors during upgrade in some GsObjectSecurityPolicies configurations

When upgrading from a version earlier than 3.5, if the GemStone base classes in the originating repository have an unusual distribution of classes and methods over the GsObjectSecurityPolicies, there may be security errors in upgradeImage, when CodeLibrarian performs the recompile of methods. (#49807)

ScaledDecimal literal parsing issues

Scaled Decimal literal strings not parsed by asNumber

Strings containing ScaledDecimal literals (using s, such as '3.45s6') were not parsed correctly by `CharacterCollection >> asNumber`, and returned PlusSignalingNaN. (#49842)

In comma-decimal point locale, ScaledDecimal literals did not parse

Locale can be used in GemStone to specify to use the comma as the decimal points, for `fromString:` and printing, but does not affect compiler parsing. The compiler was not able to parse ScaledDecimal literal syntax (using a period decimal point and s, such as 3.45s6) in a comma Locale. (#49822)

Cannot activate a passivated structure containing an instance of Utf8

The contents of an instance of Utf8 cannot be updated using standard methods, and so attempting to activate a passivated Utf8 failed. (#49864)

GCI and FFI related Issues

Error during GciExecute from UserAction preventing interrupt servicing

If an error occurs during a GciExecute* operation when invoked from a UserAction, the VM may be left in a state such that no further interrupts are serviced, such as AlmostOutOfMemory, sigAbort, etc. (#49699)

GciTsObjInfo.access could return undocumented value

This field is documented to return a value in the range 0..2, but could in fact return other values. (#49762)

Crash if GciTsPerform called with NULL GciErrSType *err arg

If this argument is NULL, the Gem crashed (#49765).

This bug was exposed by cases in which CByteArrays and CPointers that used by FFI could be committed, faulted out, and lose C data when faulted in (#49769); this bug is not fixed, but no longer causes a crash.

CHeader generated incorrect offsets for structs

`CHeader>>wrapperForTypeNamed:` previously computed incorrect offsets and sizes for fields in structs. The code now generates correct code for structs in which a C compiler would add alignment padding. (#49856)

After failed commit due to RC conflicts, commit allowed without abort

If a commit fails due to RC conflicts, an abort should be done before a subsequent attempt to commit. This abort was not required, so a commit was allowed, which could bypass correct transactional behavior, since the commit was based on the view with partial updates from the RC replay. (#49772)

gslist prints unnecessary extra Warning if no servers running

If no servers are running and gslist can produce no results, it printed an extra Warning line, in addition to reporting No GemStone Servers. (#49754)